# Package 'bufferscape'

August 5, 2026

**Title** Distance-Weighted Landscape Composition in Buffers Around Point Locations

**Version** 1.0.3

**Description** Characterises the environment surrounding point locations by computing land-cover composition within circular buffers directly from vector polygons, without conversion to a raster grid. For each site and each class it returns the exact surface area inside the buffer and a distance-decay weighted ``effective'' area in which the kernel is integrated over polygon geometry rather than evaluated at the polygon centroid, avoiding the large bias the centroid approximation introduces for elongated features passing close to the site. Polygons may overlap, so class areas are not constrained to sum to the buffer area. Intended for buffer-based exposure assessment and fine-scale spatial epidemiology, where the relevant scale is tens of metres and global land-cover products are too coarse: land-use regression around air-quality monitors, green space around residential addresses, vector-surveillance traps, and comparable designs. The classification dictionary is user-supplied, and point features and distances to off-buffer reference features are recorded alongside the areas.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 4.1)

**Imports** sf (>= 1.0.0), dplyr, tidyr, stringr, purrr, tibble, grDevices, stats, utils, tools

**Suggests** ggplot2, writexl, readxl, terra, maptiles, testthat (>= 3.0.0), knitr, rmarkdown

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**RoxygenNote** 7.3.1

**URL** https://github.com/mplanta-lab/bufferscape

**BugReports** https://github.com/mplanta-lab/bufferscape/issues

**NeedsCompilation** no

**Author** Michele Planta [aut, cre] (ORCID:
<https://orcid.org/0009-0000-3353-5368>)

**Maintainer** Michele Planta <micheleplanta@outlook.it>

**Repository** CRAN

**Date/Publication** 2026-08-05 08:10:02 UTC

# Contents



---

`batch_composition` *Process every KML in a folder*

---

## Description

Runs `buffer_composition()` over each KML in a directory, writes one workbook containing every table, one map per sampling site, and the composition and container charts.

## Usage

```
batch_composition(
  kml_dir,
  out_dir = file.path(kml_dir, "Results"),
  pattern = "\\.kml$",
  make_maps = TRUE,
  make_charts = TRUE,
```

```
  basemap = "none",
  map_zoom = NA,
  radii = c(20, 30, 40, 50),
  kernel = "exponential",
  lambda = 45,
  grid_res = 1,
  epsg = 31983,
  secondary = "weighted",
  sec_weight = 0.3,
  categories = NULL,
  palette = "aerial",
  metrics = c("exact", "weighted", "centroid", "distance"),
  label_ids = 0
)
```

## Arguments

| | |
|---|---|
| `kml_dir` | Folder holding the KML files. |
| `out_dir` | Output folder; defaults to `Results/` inside `kml_dir`. |
| `pattern` | Regular expression selecting files. |
| `make_maps`, `make_charts` | Produce figures. |
| `basemap`, `map_zoom` | Passed to the map functions. |
| `radii`, `kernel`, `lambda`, `grid_res`, `epsg`, `secondary`, `sec_weight` | Passed to `buffer_composition()`. |
| `categories` | A category dictionary; see `class_dictionary()`. |
| `palette` | Passed to `map_composition()`; `"aerial"` (default), `"colorblind"`, `"viridis"`, `"greyscale"`, or a palette table. |
| `metrics` | Which surface metrics the workbook should carry; see `write_composition_report()`. |
| `label_ids` | Print the class id on this many of the largest polygons in each map. |

## Details

A KML may hold one, two or three sampling points. Buffer polygons are named per point (`VP_9_Buffer`), while land-cover polygons and water containers are a shared pool for the whole file; each point takes its own elements **geometrically**, against its own buffer, so per-site outputs never contaminate one another.

One malformed file does not stop the batch: it is logged and skipped. Sites whose name appears in more than one file are reported in a `duplicate_traps` sheet, and their figures are given distinct filenames.

## Value

Invisibly, a list with `wide`, `long`, `tanks`, `distances`, `qc`, `summary`, `out_dir` and `failed`.

## Examples

```
dir.create(d <- tempfile())
file.copy(system.file("extdata", "example_site.kml", package = "bufferscape"), d)
out <- batch_composition(d, radii = 50, grid_res = 5,
                     make_maps = FALSE, make_charts = FALSE)
out$summary
```

---

`buffer_composition` *Buffer-level landscape metrics for point sampling sites*

---

## Description

Reads one or more KML files, and for every sampling point returns the land-cover composition of a circular buffer around it, computed directly from vector polygons.

## Usage

```
buffer_composition(
  kml,
  categories = NULL,
  driver = "KML",
  epsg = 31983,
  radii = 50,
  kernel = "exponential",
  lambda = 45,
  grid_res = 1,
  secondary = c("weighted", "primary", "both"),
  secondary_weight = 0.3,
  trap_pattern = "^[A-Za-z]{2,}[_ -]?\\d+$",
  tank_pattern = "tank|caixa|reservat",
  tank_open_pattern = "open|abert|descoberta|sem[_ ]?tampa",
  pool_pattern = "pool|piscina|swimming",
  check_kml_buffer = TRUE,
  out_xlsx = NULL,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `kml` | Path, or vector of paths, to KML files. |
| `categories` | A category dictionary. `NULL` (default) uses the built-in mare_categories schema; otherwise a `data.frame` or a path to an `.xlsx`/`.csv` file. See `class_dictionary()`. |
| `driver` | GDAL driver. Leave as `"KML"` – see Details. |
| `epsg` | EPSG code of a **projected** CRS in metres. Default 31983 (SIRGAS 2000 / UTM 23S), appropriate for Rio de Janeiro. |

`radii` Buffer radii in metres. The largest is treated as primary.

`kernel`, `lambda` Distance-decay kernel and its scale; see `decay_kernel()`.

`grid_res` Resolution in metres of the grid used to integrate the kernel over polygon geometry. Smaller is more exact and slower; 1 m is ample for a 50 m buffer.

`secondary` How to treat a hyphenated code such as `"8-6"`: `"weighted"` gives the primary class 1 - `secondary_weight` of the area and the secondary class the rest; `"primary"` ignores the secondary code; `"both"` gives the full area to each, so areas double-count deliberately.

`secondary_weight` Share of the area given to the secondary class under `secondary = "weighted"`.

`trap_pattern`, `tank_pattern`, `tank_open_pattern`, `pool_pattern` Regular expressions matching placemark names for sampling points, water tanks, the unsealed subset of tanks, and swimming pools. Pools are tested first, so a name like `"SwimmingPool"` is never mistaken for a tank. The default `trap_pattern` matches a letter prefix followed by a number (`SITE_1`, `VP_9`, `NH_12`, `trap 3`); tighten it if your point layer uses names that could collide with other placemarks.

`check_kml_buffer` If `TRUE`, cross-check any `*_Buffer` polygon in the file against the computed buffer and report its radius.

`out_xlsx` Optional path; writes every table to a workbook.

`verbose` Print progress.

## Value

A list with elements `wide` (one row per site, modelling-ready), `long` (site x category x radius), `tanks`, `distances`, `qc`, `unmatched_ids`, `meta`, `categories`, and the projected `sf` layers `traps`, `polygons`, `tanks_sf`, `pools_sf`, `lines`, `buffer_kml`.

## Overlapping polygons

Polygons may overlap – a tree crown over a roof is both – so category areas are **not** constrained to sum to the buffer area and proportions are not computed. A buffer can exceed 100% classified.

## Geometry-integrated decay

The weighted area `area_w` integrates the kernel over each polygon on a regular grid, then rescales by the exact `sf::st_area()`. Evaluating the kernel at the polygon centroid instead – reported as `area_w_centroid` for comparison – is badly biased for elongated features passing close to the sampling point, whose centroid sits at $d \approx 0$ while most of their area does not. On Maré data the bias reaches 45% for an internal road, while compact roofs stay under 1%.

## Why the KML driver

GDAL defaults to LIBKML, which overrides `<name>` with an ExtendedData field of the same name. A buffer polygon named `VP_21_Buffer` then reads as `VP_21`, is no longer recognised as reference geometry, and silently enters the land-cover totals as an unclassified polygon. Forcing `driver = "KML"` reads `<name>` correctly.

### Nothing is dropped silently

Unparseable descriptions, category codes absent from the dictionary, and points matching no pattern all raise warnings and are reported in the `qc` and `unmatched_ids` tables. Distances that were not measured stay `NA` and are never coerced to zero.

### See Also

`batch_composition()` to process a folder, `class_dictionary()` for custom dictionaries.

### Examples

```
kml <- system.file("extdata", "example_site.kml", package = "bufferscape")
res <- buffer_composition(kml, radii = 50, grid_res = 5, verbose = FALSE)
res$tanks
head(res$long[res$long$area_m2 > 0, c("full_name", "area_m2", "area_w")])
```

---

## centroid_bias *Quantify the centroid approximation bias, polygon by polygon*

---

### Description

Recomputes, for every polygon intersecting every buffer, the distance-decay weight two ways – integrated over the polygon's geometry, and evaluated at its centroid – and reports the discrepancy alongside shape descriptors.

### Usage

```
centroid_bias(
  kml,
  categories = NULL,
  driver = "KML",
  epsg = 31983,
  radii = 50,
  kernel = "exponential",
  lambda = 45,
  grid_res = 1,
  compact_cut = 0.4,
  trap_pattern = "^[A-Za-z]{2,}[_ -]?\\d+$",
  verbose = TRUE
)
```

### Arguments

| | |
|---|---|
| `kml` | Path, or vector of paths, to KML files. |
| `categories` | A category dictionary. `NULL` (default) uses the built-in mare_categories schema; otherwise a `data.frame` or a path to an `.xlsx`/`.csv` file. See `class_dictionary()`. |

| | |
|---|---|
| `driver` | GDAL driver. Leave as `"KML"` – see Details. |
| `epsg` | EPSG code of a **projected** CRS in metres. Default 31983 (SIRGAS 2000 / UTM 23S), appropriate for Rio de Janeiro. |
| `radii` | Buffer radii in metres. The largest is treated as primary. |
| `kernel`, `lambda` | Distance-decay kernel and its scale; see `decay_kernel()`. |
| `grid_res` | Resolution in metres of the grid used to integrate the kernel over polygon geometry. Smaller is more exact and slower; 1 m is ample for a 50 m buffer. |
| `compact_cut` | Compactness below which a polygon is called elongated. |
| `trap_pattern` | Regular expression matching the placemark names of sampling points. |
| `verbose` | Print progress. |

## Details

This is the diagnostic behind the claim that centroid weighting is unsafe. The bias is not uniform: it is negligible for compact features and large for elongated ones that pass close to the site, because such a polygon's centroid can sit almost on the sampling point while most of its area does not.

## Value

A tibble, one row per polygon per site per radius:

`Ovitrap_ID`, `radius_m`, `source_file` identifiers

`id_primary`, `label_en` class

`area_m2` clipped area inside the buffer

`d_centroid` distance from the site to the polygon centroid (m)

`d_min`, `d_max` nearest and farthest distance from the site to the polygon (m)

`centroid_inside` `FALSE` when the polygon's centroid falls outside the polygon, as happens for concave features. The centroid weight is then evaluated at a point that is not part of the feature.

`compactness` $4\pi A/P^2$; 1 for a circle, towards 0 as the outline becomes elongated or convoluted

`d_mean` area-weighted mean distance from the site to the polygon (m). This is the quantity the centroid is standing in for.

`centroid_offset` `(d_mean - d_centroid) / lambda`; how far the centroid sits from the polygon's mean distance, in kernel scale lengths. This is the mechanistic driver of the bias: a polygon whose centroid is much nearer than its typical point gets over-weighted.

`reach_ratio` `(d_max - d_min) / (d_centroid + 1)`; the span of the polygon relative to its centroid distance.

`geometry_class` `"compact"` or `"elongated"`, split at `compact_cut`

`w_integrated`, `w_centroid` mean kernel weight, both ways

`area_w`, `area_w_centroid` weighted areas

`w_nearest`, `area_w_nearest`, `bias_nearest_pct` the same, for a weight taken at the polygon's nearest point. Included because it is the obvious alternative to the centroid; it is an upper bound on the mean weight and overestimates more than the centroid does, on both compact and elongated polygons.

`bias_pct` $100(area\_w\_centroid - area\_w)/area\_w$

## See Also

summarise_centroid_bias()

## Examples

```
kml <- system.file("extdata", "example_site.kml", package = "bufferscape")
b <- centroid_bias(kml, radii = 50, grid_res = 5)
summarise_centroid_bias(b)
```

---

`class_dictionary` *Land-cover category dictionary*

---

## Description

A dictionary maps the numeric codes written in the <description> field of each digitised polygon to a named land-cover category. Everything downstream – the area columns, the map palette, the legend labels – is driven by this table, so the workflow is not tied to any one classification scheme.

## Usage

```
class_dictionary(x = NULL)
```

## Arguments

| | |
|---|---|
| x | One of:<br>• NULL (default) – the built-in mare_categories schema.<br>• a data.frame – validated and used as is.<br>• a path to an .xlsx or .csv file – read, then validated. |

## Details

The dictionary shipped with the package, mare_categories, is the 29-class schema developed for the Complexo da Maré ovitrap study. Supply your own to work in a different setting.

A dictionary must contain:

`id` integer, unique, positive. The code written in the KML <description> field.

`category` character. A coarse grouping (for example "cobertura", "vegetacao"). Used for aggregation and for column names.

`description` character. The specific class within the grouping.

and may optionally contain:

`label_en` character. The label printed on maps, charts and legends. Defaults to `category description`.

`fill` character. A hex colour for the map. If absent, a colour is generated. Supplying your own is strongly preferred for land cover, because a generated ramp assigns arbitrary hues that carry no meaning.

`pattern` character, one of `"none"`, `"dots"`, `"dots2"`, `"diag"`, `"diag2"`, `"cross"`, `"horiz"`, `"vert"`, `"grid"`. Overprints a texture so that classes sharing a base colour stay distinguishable. Defaults to `"none"`.

A `full_name` column (`id_category_description`) is added automatically and is what the output column names are built from.

## Value

A `data.frame` with columns `id`, `category`, `description`, `label_en`, `fill`, `pattern` and `full_name`.

## See Also

mare_categories for the built-in schema.

## Examples

```
# the built-in 29-class schema
head(class_dictionary())

# a minimal custom dictionary
own <- data.frame(
  id          = 1:3,
  category    = c("water", "built", "vegetation"),
  description = c("pond", "roof", "canopy"),
  fill        = c("#2C7FB8", "#BDBDBD", "#31A354")
)
class_dictionary(own)
```

---

# class_palette *Colour and texture schemes for a dictionary*

---

## Description

Turns a class dictionary into the `fill` and `pattern` used by the map and chart functions. Separating this from the drawing code means a scheme can be inspected, tested and swapped without touching a figure.

## Usage

```
class_palette(
  categories = class_dictionary(),
  scheme = c("aerial", "colorblind", "viridis", "greyscale"),
  hues = NULL
)
```

## Arguments

| | |
|---|---|
| categories | A dictionary, as returned by class_dictionary(). |
| scheme | One of: |
| | "aerial" (default) the dictionary's own fill and pattern. For mare_categories these echo the appearance in aerial imagery: greens for vegetation, terracotta for clay tile, near-white for reflective metal, greys for concrete, rust for corroded metal. |
| | "colorblind" one colour-vision-safe hue per coarse group, with members of a group separated by a lightness step **and** a texture. See Details. |
| | "viridis" the viridis ramp ordered by id, no textures. Useful when the classes are ordinal rather than nominal. |
| | "greyscale" a lightness ramp plus textures, for print or photocopying. |
| hues | Optional character vector of hex colours to use as the group hues under scheme = "colorblind", overriding the built-in bank. |

## Value

A data.frame with columns id, fill and pattern.

## Why "colorblind" works by group

A palette of 29 nominal colours cannot be made safe for colour-vision deficiency; the perceptual space is not large enough, and any such palette will contain pairs that converge under deuteranopia. Measured on mare_categories, even a palette built entirely from colour-vision-safe primaries leaves a minimum pairwise distance of roughly 4 under simulated deuteranopia, below a comfortable discrimination threshold.

The "colorblind" scheme therefore stops asking colour to do the whole job. Colour distinguishes the **coarse group** only – 12 of them for mare_categories, where the minimum distance under simulated deuteranopia rises to about 7.6 – and within a group, members are separated by a lightness step and by a distinct texture. No class depends on hue as its only cue, which is the property that makes a figure accessible, and it also survives greyscale printing.

If a dictionary has more groups than available hues, the bank is cycled with a lightness offset and a warning is issued: at that point the grouping is probably too fine to carry by colour at all.

## See Also

class_dictionary(), map_composition()

## Examples

```
# the appearance-matched default
head(class_palette())

# colour-vision-safe alternative
head(class_palette(scheme = "colorblind"))

# with a custom dictionary
```

```
own <- data.frame(id = 1:4,
                  category    = c("water", "water", "built", "veg"),
                  description = c("pond", "channel", "roof", "canopy"))
class_palette(own, scheme = "colorblind")
```

---

`decay_kernel` *Distance-decay kernels*

---

## Description

Weights are bounded on $[0, 1]$ with $w(0) = 1$, so a weighted area is always between zero and the true area and is directly interpretable as an "effective" area.

## Usage

```
decay_kernel(d, kernel = "exponential", lambda = 45)
```

## Arguments

| | |
|---|---|
| `d` | Numeric vector of distances, in the units of the projected CRS (metres for a UTM zone). |
| `kernel` | One of `"exponential"` (default), `"gaussian"` or `"none"`. `"none"` returns 1 everywhere, giving unweighted areas. |
| `lambda` | Decay scale, in the same units as `d`. |

## Details

This matters: the ratios `area / d` and `area / d^2` sometimes used for the same purpose are not kernels. They diverge as a polygon approaches the sampling point, have no upper bound, are not normalisable, and carry uninterpretable units, so their coefficients cannot be compared across sites.

The default `lambda = 45` follows close-kin genetic estimates of mean *Aedes aegypti* dispersal (Jasper et al. 2020, *BMC Biology*, 45.2 m, 95% CI 39.7-51.3), consistent with a Brazilian mark-release-recapture estimate of 52.8 m (Winskill et al. 2015). Fine-scale dispersal is repeatedly described as exponential (Laplacian), which is why that is the default form. For another taxon or another process, set `lambda` from the relevant literature, and report a sensitivity analysis over a grid of values compared by AIC.

## Value

A numeric vector of weights the same length as `d`.

## Examples

```
decay_kernel(c(0, 25, 50), lambda = 45)
decay_kernel(c(0, 25, 50), kernel = "gaussian", lambda = 45)
```

`map_basemap` *Satellite basemap figure for one sampling site*

## Description

Imagery with the buffer and the sampling point, and nothing else.

## Usage

```
map_basemap(
  res,
  trap,
  radius = max(res$long$radius_m),
  basemap = "esri",
  zoom = NA,
  file = NULL,
  width = 8,
  height = 8,
  dpi = 300
)
```

## Arguments

| | |
|---|---|
| `res` | Output of `buffer_composition()`. |
| `trap` | Name of the sampling point to draw. |
| `radius` | Buffer radius; defaults to the largest computed. |
| `basemap` | `"esri"` for Esri World Imagery, a path to a georeferenced raster of your own, or `"none"`. |
| `zoom` | Tile zoom level; `NA` picks the finest the provider serves. |
| `file` | Optional output path. |
| `width, height, dpi` | Passed to `ggplot2::ggsave()`. |

## Details

Tiles are requested in the CRS the figure is drawn in, so the raster is never resampled and cannot drift relative to the polygons. Requires **maptiles** and **terra**; without them the panel renders white and a warning is issued rather than failing.

Google Earth imagery is not offered: its terms permit alteration only inside Google software, and it cannot be relicensed for an open-access figure.

## Value

A **ggplot** object.

`map_combined` *Combined imagery and land-cover figure for one sampling site*

## Description

Imagery, polygons and legend in a single panel.

## Usage

```
map_combined(
  res,
  trap,
  radius = max(res$long$radius_m),
  basemap = "none",
  zoom = NA,
  top_n = 9,
  poly_col = "#E02020",
  file = NULL,
  width = 13,
  height = 7.6,
  dpi = 300
)
```

## Arguments

| | |
|---|---|
| `res` | Output of `buffer_composition()`. |
| `trap` | Name of the sampling point to draw. |
| `radius` | Buffer radius; defaults to the largest computed. |
| `basemap`, `zoom` | See `map_basemap()`. |
| `top_n` | Number of classes listed individually in the legend; the rest are pooled as "other categories". |
| `poly_col` | Outline and fill colour used for every polygon. |
| `file` | Optional output path. |
| `width`, `height`, `dpi` | Passed to `ggplot2::ggsave()`. |

## Value

A **ggplot** object.

`map_composition` *Land-cover map for one sampling site*

## Description

A white-ground figure: polygons coloured by category, water containers, the buffer, and a legend panel giving the surface composition, the container counts and the distances to reference features.

## Usage

```
map_composition(
  res,
  trap,
  radius = max(res$long$radius_m),
  palette = "aerial",
  top_n = 10,
  fill_alpha = 0.55,
  patterns = TRUE,
  show_unclassified = TRUE,
  label_ids = 0,
  legend = c("composition", "containers", "key"),
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  file = NULL,
  width = 13,
  height = 8,
  dpi = 300
)
```

## Arguments

| | |
|---|---|
| `res` | Output of `buffer_composition()`. |
| `trap` | Name of the sampling point to draw. |
| `radius` | Buffer radius; defaults to the largest computed. |
| `palette` | Colours and textures. A scheme name – `"aerial"` (default), `"colorblind"`, `"viridis"` or `"greyscale"` – or a `data.frame` with `id`/`fill`/`pattern`, or a vector of colours named by class id. A partial palette is allowed: unlisted classes fall back to the dictionary's own colour. See `class_palette()`. |
| `top_n` | Number of classes listed individually in the legend; the rest are pooled as "other categories". |
| `fill_alpha` | Fill transparency. Semi-transparent fills let overlapping polygons show through, which is the point. |
| `patterns` | Draw the textures defined by the palette. `FALSE` gives flat fills. |

`show_unclassified`
: Draw and quantify the part of the buffer covered by no polygon, hatched in grey.

`label_ids`
: Print the class id on this many of the largest polygons. `0` (default) prints none. Useful with large dictionaries, where colour alone cannot separate every class.

`legend`
: Which legend sections to show: any of `"composition"`, `"containers"`, `"key"`. `FALSE` suppresses the legend entirely and the map fills the panel.

`title`, `subtitle`, `caption`
: Text overrides. `NULL` uses the automatic text; `""` suppresses that line.

`file`
: Optional output path.

`width`, `height`, `dpi`
: Passed to `ggplot2::ggsave()`.

## Details

Polygons are clipped to a square frame set outside the buffer, so features overshoot the circle as they do in the imagery but never run across the legend. Overlapping polygons are drawn largest-first, so small features stay visible. Patterns separate classes that share a base material: dots for debris, crosshatch for degraded mixed slab, diagonals for active construction, and grey hatching for unclassified ground.

## Value

A **ggplot** object, invisibly written to `file` if given.

## Customising beyond the arguments

The return value is an ordinary **ggplot** object, so anything not exposed as an argument can be added afterwards – `+ ggplot2::labs()`, `+ ggplot2::theme()`, further layers. One exception is worth knowing: the legend is drawn as text **inside the panel**, not as a ggplot guide, so `theme(legend.*)` has no effect on it. That is why `legend`, `top_n` and `title` are explicit arguments.

Drawing deliberately avoids `geom_sf()`, `coord_sf()` and composition packages. Geometry is converted to plain coordinates and drawn with `ggplot2::geom_polygon()` under `ggplot2::coord_fixed()`, which is the most version-stable combination and cannot lose layers to CRS renegotiation at render time.

## Examples

```
kml <- system.file("extdata", "example_site.kml", package = "bufferscape")
res <- buffer_composition(kml, radii = 50, grid_res = 5, verbose = FALSE)
if (requireNamespace("ggplot2", quietly = TRUE)) map_composition(res, "SITE_1")
```

`mare_categories` *Maré 29-class urban morphology dictionary*

## Description

The land-cover classification schema developed for oviposition-trap surveillance in the Complexo da Maré favela complex, Rio de Janeiro. It resolves distinctions that global land-cover products cannot at this scale: fibrocement against sealed and unsealed concrete slab, individual water containers, narrow alleys, polluted open channels, and active construction.

## Usage

```
mare_categories
```

## Format

A data frame with 29 rows and 6 columns:

**id** integer code written in the KML `<description>` field

**category** coarse grouping, in Portuguese

**description** specific class, in Portuguese

**label_en** English label used on figures

**fill** hex colour, chosen to echo the appearance in aerial imagery

**pattern** overprinted texture: `"none"`, `"dots"`, `"dots2"`, `"diag"` or `"cross"`, used where classes share a base material

## Details

The schema is a nested hierarchy – macro land use, then vegetation configuration, then building complexity – which is why `category` and `description` are separate columns.

## Source

Planta M. Qualitative assessment schema for urban polygons in Rio de Janeiro favelas, interpreted from high-resolution aerial imagery.

## Examples

```
head(mare_categories)
subset(mare_categories, pattern != "none")
```

`parse_class_codes` *Parse a polygon description into primary and secondary category codes*

## Description

Digitised polygons carry their class in the KML `<description>` field. A single code (`"7"`) is unambiguous; a hyphenated pair (`"8-6"`) records a primary and a secondary class where the interpreter was uncertain or the surface is genuinely mixed.

## Usage

```
parse_class_codes(x)
```

## Arguments

| | |
|---|---|
| `x` | Character vector of description fields. |

## Details

Naive coercion silently destroys these: `as.numeric("8-6")` is `NA`, and an inner join on `NA` drops the polygon without warning. In the Maré data that would discard roughly a fifth of all polygons, and non-randomly – the ambiguous ones are the mixed-material and transitional surfaces.

## Value

A tibble with `id_primary`, `id_secondary` and `n_codes`.

## Examples

```
parse_class_codes(c("7", " 6 ", "8-6", "5 / 7", "", NA))
```

`plot_group_points` *Water containers per site across a community*

## Description

One stacked column per sampling site: sealed tanks, unsealed tanks and swimming pools, so that the abundance and the nature of the larval habitat read together at a glance.

### Usage

```
plot_group_points(
  tanks,
  community,
  radius = max(tanks$radius_m),
  file = NULL,
  width = 11,
  height = 6,
  dpi = 300
)
```

### Arguments

| | |
|---|---|
| tanks | The tanks table from buffer_composition() or batch_composition(). |
| community | Value of the site-name prefix to plot, for example "VP". |
| radius | Buffer radius; defaults to the largest present. |
| file | Optional output path. |
| width, height, dpi | Passed to ggplot2::ggsave(). |

### Details

Segments are ordered by how available the water is to an ovipositing female – sealed, then unsealed, then open pool – on a viridis ramp. Sealed tanks usually dominate by an order of magnitude, so a single unsealed tank would otherwise be a one-pixel sliver: white separators keep every segment legible and the oviposition-available count is called out above each column.

### Value

A **ggplot** object.

---

## plot_site_composition *Land-cover composition chart for one sampling site*

---

### Description

Horizontal bars, one per category present in the buffer, ordered by area. Categories absent from the buffer are omitted, so the panel always fits whatever the size of the dictionary.

### Usage

```
plot_site_composition(
  res,
  trap,
  radius = max(res$long$radius_m),
  palette = "viridis",
```

```
  file = NULL,
  width = 9,
  height = 6,
  dpi = 300
)
```

## Arguments

| | |
|---|---|
| `res` | Output of `buffer_composition()`. |
| `trap` | Name of the sampling point. |
| `radius` | Buffer radius; defaults to the largest computed. |
| `palette` | `"viridis"` (default) orders hue by magnitude. Any other value is passed to `resolve_palette()` – `"aerial"` makes the bars match the colours used by `map_composition()`, `"colorblind"` matches the colour-vision-safe scheme. |
| `file` | Optional output path. |
| `width, height, dpi` | Passed to `ggplot2::ggsave()`. |

## Value

A **ggplot** object.

## Examples

```
kml <- system.file("extdata", "example_site.kml", package = "bufferscape")
res <- buffer_composition(kml, radii = 50, grid_res = 5, verbose = FALSE)
if (requireNamespace("ggplot2", quietly = TRUE))
  plot_site_composition(res, "SITE_1")
```

---

# `resolve_palette` *Resolve a palette argument*

---

## Description

Accepts a scheme name, a data frame, or a named vector of colours, and returns a canonical palette table. Used internally by the figure functions so they all take `palette` the same way.

## Usage

```
resolve_palette(palette = "aerial", categories = class_dictionary())
```

## Arguments

| | |
|---|---|
| `palette` | A scheme name, a `data.frame` with `id`/`fill` (and optionally `pattern`), or a vector of colours named by class `id`. |
| `categories` | The dictionary in use. |

## Value

A `data.frame` with `id`, `fill`, `pattern`.

## Examples

```
resolve_palette("colorblind")
resolve_palette(c("1" = "#FF0000", "2" = "#00FF00"))
```

---

`summarise_centroid_bias` *Summarise centroid bias*

---

## Description

Collapses `centroid_bias()` output into the table a methods paper needs: the distribution of the discrepancy, split by polygon geometry, and the worst offenders by class.

## Usage

```
summarise_centroid_bias(
  x,
  by = c("geometry", "class", "site"),
  area_weighted = FALSE
)
```

## Arguments

| | |
|---|---|
| `x` | Output of `centroid_bias()`. |
| `by` | Grouping: `"geometry"` (default), `"class"` or `"site"`. |
| `area_weighted` | If `TRUE`, weight each polygon's bias by its area, so the summary reflects the effect on the covariate a model would actually see rather than treating a 2 m2 sliver like a 2000 m2 block. |

## Value

A tibble.

## Examples

```
kml <- system.file("extdata", "example_site.kml", package = "bufferscape")
b <- centroid_bias(kml, radii = 50, grid_res = 5)
summarise_centroid_bias(b, by = "class")
```

## validate_dictionary *Validate and complete a category dictionary*

### Description

Checks the required columns, fills in the optional ones, and returns a dictionary in canonical form. Exported so a custom dictionary can be checked before a long batch run rather than failing partway through.

### Usage

```
validate_dictionary(x)
```

### Arguments

| | |
|---|---|
| x | A data.frame. |

### Value

The completed dictionary.

### Examples

```
validate_dictionary(
  data.frame(id = 1:2, category = c("a", "b"), description = c("x", "y"))
)
```

## viridis_colours *Viridis colours without a package dependency*

### Description

Anchor colours sampled from the viridis colormap, interpolated with grDevices::colorRampPalette(). Avoids depending on **viridisLite** for what is a handful of hex codes.

### Usage

```
viridis_colours(n)
```

### Arguments

| | |
|---|---|
| n | Number of colours. |

### Value

A character vector of n hex colours.

## Examples

```
viridis_colours(5)
```

---

`write_composition_report` *Write a composition workbook*

---

## Description

Exports the tables from `buffer_composition()` or `batch_composition()` to a multi-sheet `.xlsx` file, with control over which radii, which surface metrics and which sheets are included.

## Usage

```
write_composition_report(
  x,
  path,
  radii = NULL,
  metrics = c("exact", "weighted", "centroid", "distance"),
  per_radius = TRUE,
  combined = TRUE,
  long = TRUE,
  extras = TRUE,
  key = TRUE,
  labels = c("full_name", "label_en"),
  digits = NULL
)
```

## Arguments

| | |
|---|---|
| `x` | The list returned by `buffer_composition()` or `batch_composition()`. |
| `path` | Output `.xlsx` path. |
| `radii` | Radii to include. `NULL` (default) includes every radius present. The largest is treated as primary and its sheet is written first. |
| `metrics` | Which per-class surface metrics to export, any of:<br>`"exact"` `_area_m2`, the exact area inside the buffer<br>`"weighted"` `_area_w`, kernel integrated over polygon geometry – the metric intended for modelling<br>`"centroid"` `_area_w_centroid`, kernel at the polygon centroid, for methods comparison<br>`"distance"` `_d_nearest_m`, distance to the nearest patch of the class, `NA` when absent<br>Defaults to all four. Dropping `"centroid"` roughly halves the width of a wide sheet if the methods comparison is not needed. |

| | |
|---|---|
| per_radius | Write one modelling-ready sheet per radius (wide_50m, wide_30m, ...). |
| combined | Write the single wide sheet holding every radius at once. |
| long | Write the tidy site x class x radius table. |
| extras | Write the supporting sheets: tanks, distances, qc, categories, settings, and any diagnostic sheets present (unmatched_ids, failed_files, duplicate_traps, buffer_check). |
| key | Write the metric_key sheet explaining every column suffix. |
| labels | Class naming in the wide sheets: "full_name" (default, id_category_description, stable and machine-friendly) or "label_en" (readable, but changes if the dictionary's labels are edited). |
| digits | Round numeric columns to this many decimals. NULL leaves full precision. |

## Details

Exposed as its own function so that a workbook can be produced from a single buffer_composition() call, re-exported with different content without recomputing, or trimmed before sharing.

## Value

Invisibly, the named list of data frames that was written.

## See Also

buffer_composition(), batch_composition()

## Examples

```
kml <- system.file("extdata", "example_site.kml", package = "bufferscape")
res <- buffer_composition(kml, radii = c(30, 50), grid_res = 5,
                          verbose = FALSE)
out <- file.path(tempdir(), "composition.xlsx")
if (requireNamespace("writexl", quietly = TRUE))
  write_composition_report(res, out, metrics = c("exact", "weighted"))
```

# Index